FROM A GRAPH TO A DEVELOPMENT CYCLE: MBSE AS AN APPROACH TO REDUCE DEVELOPMENT EFFORTS

M. Obstbaum, U. Wurstbauer, C. König, T. Wagner, C. Kübler, V. Fäßler,
TWT GmbH Science & Innovation, Erni Singerl Straße 2, 85053 Ingolstadt, Germany

Abstract

The development of complex systems, such as airplanes, spacecraft or automobiles, requires sophisticated and innovative approaches to reduce cost and time of the development cycle and to be able to cope with the challenges that new concepts and architectures pose. Especially the continuity of methods and tools is often missing between the different stages of the development cycle. Various barriers need to be overcome for an initial idea to be transferred from requirements definition and functional development to first testing and down to the final product. One suitable approach to lower these barriers and thus reduce development costs and efforts is the application of model-based systems engineering (MBSE). This allows the collection, combination and automatic further processing of information from SysML-graphs to requirements and even first functional prototypes.

1. PRESSURE TO BE INNOVATIVE IN TIMES OF COMPLEX SYSTEMS AND SHORT TIME-TO-MARKET PERIODS

The market demands products and services that are innovative at ever shorter time-to-market periods. Driven by the software application industry with its short and innovative development cycles and agile project management methods the evolution of ambitious customer demands reaches out to the realms of automotive and aerospace industry. It is essential for these industries to gain the needed dynamic robustness as well as the required drive for evolutionary and disruptive innovations. Affected fields are e.g. electrification of power units, connected services, functions in terms of assistant and autonomous mobility as well as real-time data acquisition and data processing/supervision. To this end, the already complex automobiles, aircraft and spacecraft are reaching ever-higher levels of complexity. This has to be accounted for in order to keep and further improve the high quality levels of products and services.

One of the primary dedications of systems engineering (SE) is to manage and diminish complexity [1, 2]. Therefore, especially in the aerospace domain with its foremost interest in building reliable, safe and secure systems it is widely spread ever since. However, in the conflict of high complexity and short time-to-market periods it is essential to improve methods and concepts further. One of the most promising approaches is Model Based Systems Engineering (MBSE) which at its heart synthesizes the artefacts emerging in the specification and design process in a graph. A graph is a network of relations and in MBSE it is used to create ontologies. They are typically created using graphical modeling languages (like UML in software engineering) and the so-called model tree collects the underlying machine readable and interpretable relational data structure. It includes meta-information and provides the possibility of automation of essential aspects of the development cycle. This said, the disciplines of requirements engineering, specification and design, simulation, virtualization and testing are merging into a holistic framework that allows for context adjusted workflow approaches be it top-down, bottom-up, iterative, waterfall-dominated or agile. Furthermore, the graph-like specification and design basis reaches out to all of the phases of a system's lifecycle.

The aforementioned issues are addressed in detail in the following paragraphs taking into account aspects of applicable methodologies and the use of open standards like the SysML (www.omgsysml.org), Open Services for Lifecycle Collaboration OSLC (https://open-services.net), and the Functional Mock-up Interface FMI (www.fmi-standard.org). A special focus is put on the graph based tailoring of development cycles. Finally, the theoretical concepts will be demonstrated in the tool chain examples of the INTO-CPS project.

2. DEVELOPMENT CYCLES AND THE AUGMENTED V-MODEL

Modern development cycles require agile collaboration, which does not fit to waterfall-like project management. Software projects have proven that planning should be permeable with respect to the continuous update of requirements. Considering the evolving possibilities of (co-)simulation and virtual product development this becomes increasingly true also for general technical systems and especially in the automotive and aerospace industry. A framework for developing systems should allow for iteration, incrementalism, and recursion as well as the application of agile systems engineering methods. In this paper, the concept of the V-Model [3] is the basis in terms of evaluation and refinement of workflows in development cycles by putting it in the context of MBSE and graphical modelling. The adapted version of the V- Model used here is presented

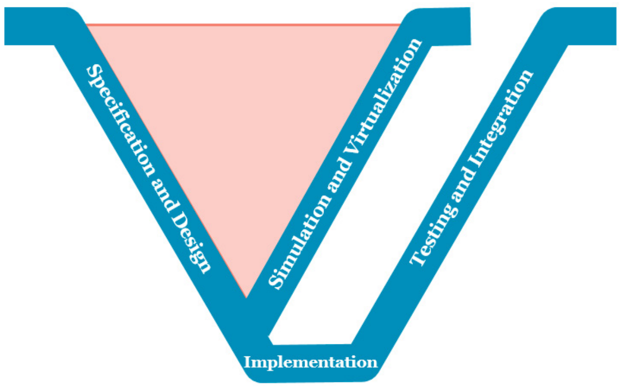

Figure 1: The augmented V-Model as the workflow model for a generic development cycle

in Figure 1. In addition to the activities of "Specification and Design", "Implementation" and "Testing and Integration" the V-Model is augmented with a parallel bottom-up-branch to account for the activities of "Simulation and Virtualization". The red-colored triangle shows the part of the development process where the use of a graph is most advantageous. Here, the graph is created and fostered in the "Specification and Design" activity and allows for continuous verification and validation in simulation and virtualization without breaking the single-source-of-truth approach. The role of the graph in development cycles, be they vintage or agile, is further elaborated in Figure 2. Starting from the top left of the augmented V-Model with setting up an adequate framework in terms of both organizational aspects as well as technological premises is implemented.

Creating, collecting and, managing requirements, leads to the initiation of MBSE and at its heart the synthesis of graph ontology. The graph with its machine readable and interpretable data structure is used as the basis for all relevant activities within the development cycle. It is also iteratively feeding back to requirements engineering. The graph links detailed specification and design of technical components, such as mechatronic, software or persistent data.

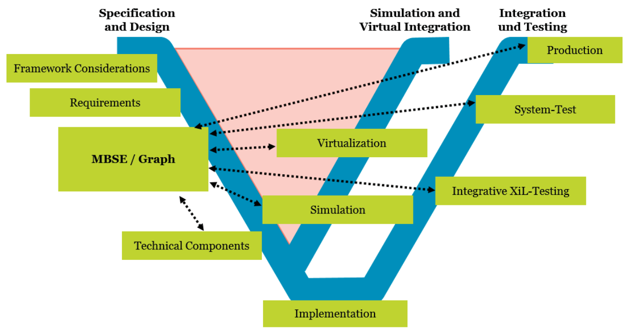

Figure 2: The augmented V-Model and the activities of a generic development cycle.

Furthermore, the augmented V-Model allows for automated generation of simulation models and the virtualization of their functionalities. It offers a consistent fast track from conceptual design to functional prototyping. This can for instance result in virtual test flights or virtual test driving. Nowadays virtualization frameworks are based on gaming engines to offer unprecedented possibilities of realistic virtual testing.

The graphs are also the basis for the specification of integrative XiL testing, such as model, software, processor, or hardware in the loop tests. Model in the loop (MiL) tests are executed efficiently when having a direct link between simulation models and its corresponding loop. The same is true for hardware in the loop (HiL) and software in the loop (SiL). SiL can be implemented through virtual electric control units, where the specification, the design and the loop are linked through a graph.
Real and virtual system testing is completely continuous on the basis of graph specification and design. There is the possibility of going back and forth between virtual and real systems. Furthermore, it is possible to have parallel paths of the two worlds, picking out the best of both with respect to efficiency and costs for each individual step in development process. The integration of real and virtual system testing is a key element in order to build safe and secure automated or autonomous vehicles.
Finally, the graph as the basis of a system specification integrates the development into the production domain. In this respect the graph is the blue print for the production of the system which, so to speak, knows by itself the "WHAT" and "HOW" of the needed production steps.

3. APPLICATION OF MBSE AND GRAPH ONTOLOGIES

The application of MBSE as a method of SE comprises several methodological concepts. To this end it is very important to use an adequate method, modeling language and tool which support each other and the advantages of MBSE paradigms. The paradigms are object oriented analysis and design, focus on systems functionality in the development process as well as automatization and deduction of aspects from graph ontologies.

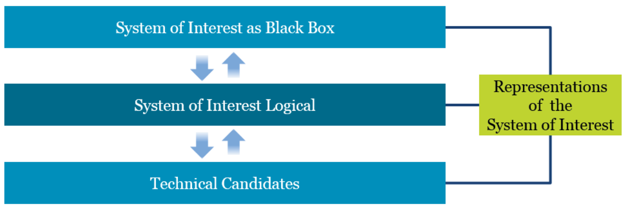
System of Interest as Black Box
System of Interest Logical
Technical Candidates
Representations of the System of Interest

Figure 3: Representations of the System of Interest cf. OOSEM and indication of permeability between the different abstractions.

The original methodological concept of MBSE assumes three different abstractions of the system of interest. This can be identified as the collective property of four of the most prominent methods in MBSE, which are: the Object-oriented Systems Engineering Method (OOSEM) [4], Harmony/SE [5], the Systems Modeling Toolbox SysMOD [6] and the Functional Architecture of Systems (FAS) [7]. However, the strongest conceptual impact in the context at hand is clearly made by OOSEM. The separation of abstraction layers is a key to manage variation and to diminish complexity by adding an inherent traceability through the use of a graph. As shown in Figure 3 the representations of the system of interest (SoI) are the SoI as Black Box, the SoI Logical and the Technical Candidates. In the following, the evolution of a holistic graph, based on these representations of the SoI is elaborated.

The name SoI as Black Box derives as a viewpoint of the SoI's context elements, viz. its stakeholders. It is crucial to put the SoI in a specific context in order to uniquely define the borders of the SoI and to be able to address valid stakeholder needs, use cases as well as relevant use case scenarios. The properties of the SoI as Black Box which result from the analysis of stakeholder needs, use cases and use case scenarios, are the measures of effectiveness and performance plus the system functions composing the functionality represented by the use cases.

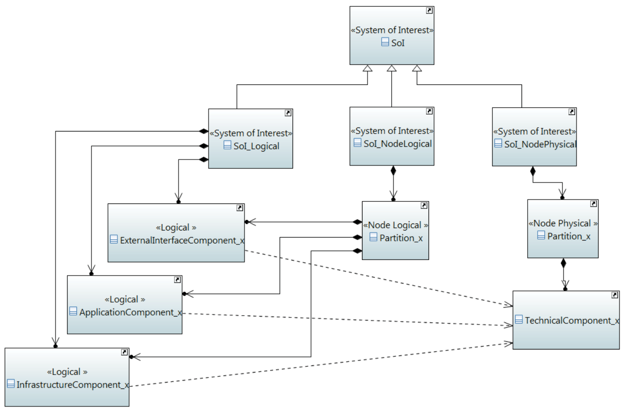

Figure 4: The abstractions of the System of Interest (SoI) represented by a graph ontology.

These properties are the input for the next abstraction layer where the logical specification and design of the SoI takes place. Figure 4 shows how the object-oriented paradigm implemented in SysML creates the graph. The SoI_Logical represents a specialization of the SoI as Black Box, the latter is named SoI in the graph. It is the root of the graph and is placed at the top of the diagram canvas. Using the SysML's generalization relation the SoI_Logical inherits the properties of the SoI in order to detail them with respect to functional and logical decomposition. Logical decomposition is used to identify logical components and their functional aspects. The model element used here for decomposition is a directed part association, with the filled diamond symbol at one association end and the arrow symbol, which indicates the part at the other end. The tentative components of the SoI_Logical are herein classified as external interface components, application components (this is where the business logic of the SoI_Logical is elaborated) and infrastructure components. Putting the letter x at the end of the names of the respective model elements in Figure 4 simply indicates that it could be any logical component of this type. While one can make a different classification with respect to principle logical elements, it is essential to keep the functional / logical component independent of concrete technical solutions. It is about the creation of functional / logical basis which is consistent with the identified use cases and enables the selection of the optimal technical solution candidate for current circumstances. The corresponding modelling and analysis results are usually synthesized in the logical architecture, which specifies details of ports and flows between logical components. However, the concept of logical architecture shall not be further detailed here.

The input towards specification and design of possible technical candidates are the logical components and their properties. An important step towards technical candidates of the SoI is to analyze and model the partitioning of logical components upon nodes. The identification of nodes takes place against considerations of physical location, technological framework and measures of performance. In the graph representation of Figure 4 this principle corresponds to the SoI_NodeLogical being a specialization of the SoI and being composed by the identified partitions as parts. The node logical partitions are composed of distinct logical components. The possible technical candidates are then modeled by allocating the properties of logical to distinct technical components, viz. hardware, software and persistent data. For this purpose the allocate relationship between the logical and technical components, represented by the dashed arrows in Figure 4 is used. The graph ontology is completed by composing the partitions of the SoI_NodePhysical by the identified technical components. The SoI_NodePhysical is a specialization of the SoI.

A few things are important to notice in terms of the graph ontology displayed in Figure 4. Every path between the elements of the path can be traced and used for verification in terms of coherent form and content using model checks. On each layer of the graph's hierarchal structure all elements can be used as variation points while keeping the traceability of aspects. Last but not least simulation and virtualization activities can be used continuously to each hierarchy level and element or holistically to the whole graph.

4. CONTINUITY IN PROCESSES AND TOOLS

To demonstrate the real-life application of the MBSE approach, discussed in the previous sections, the implementation in a tool chain is presented in this section.

The aim of the INTO-CPS project [8] is to provide an Integrated Toolchain for model-based design of Cyber-Physical Systems (CPSs). The different regions of the V-cycle are represented by several specialized tools. However, one key aspect of INTO-CPS is the openness of the tool-chain, so that the tool-chain can be extended, or single tools can be replaced with alternative tools that comply with the standards that are described below. This takes into account the fact that each use-case for MBSE is slightly different, and organizations may have legacy tools that they need to keep.

The abstract specification of the system is done in SysML, where a specific profile for INTO-CPS was created, suited specifically for the design of CPS. Here, the Modelio tool (www.modelio.org) is used for SysML modelling and requirements definition. While several diagrams are created for the INTO-CPS profile, two are of prime importance from the tool-chain point of view.

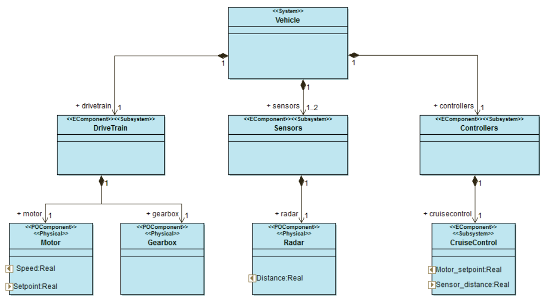

Figure 5: an Architecture Diagram of a simplified vehicle, using the INTO-CPS profile for SysML

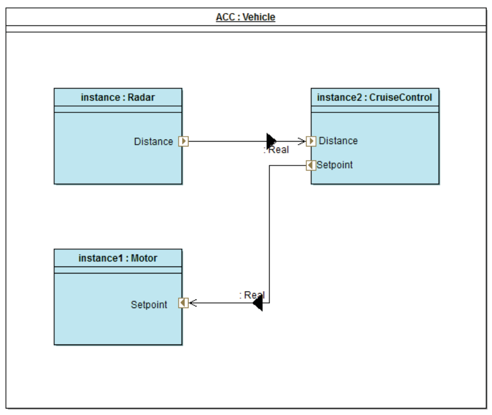

Figure 6: a Connections Diagram of a simple cruise control system using the INTO-CPS profile for SysML

The architecture diagram (see Figure 5) represents the SoI on different levels of abstractions, down to single blocks with flow ports (where flow is in this context a signal that is exchanged in the following Co-simulation) and parameters. The architecture diagram only represents the system architecture with its constituent parts on an abstract level. A specific implementation is described in a connections diagram, where the flow of signals between the different system parts is described. Such a connections diagram is shown in Figure 6. A crucial step for the tool-chain aspect is the export from SysML to FMI. For example, a SysML block of a sub-system already contains all the relevant information to describe the interface of the sub-system according to the FMI standard. Furthermore, a complete connections diagram already describes the flow of signals between models in a Co-simulation scenario.

To add the logical or physical behavior of the sub-systems to the abstract interface definition, the interface (described in the FMI standard) is then imported into modeling tools that are suited for the specific modeling task. In the context of INTO-CPS, these are OpenModelica (www.openmodelica.org), 20-sim (www.20-sim.com) and Overture (www.overturetool.org). The logical behavior of a controller can be modelled in Overture for example, while the physics of the surrounding system can be modeled in 20-sim or OpenModelica.

When the different parts of the complete system are modeled on their own, or imported from model libraries, they are exported from the modeling tools to a Functional Mock-up Unit (FMU). An FMU uses the FMI as interface and can either contain executable models or serve as a wrapper to control the stand-alone simulation tool. The models are then simulated together in a Co-simulation setting. In INTO-CPS a Co-Simulation Orchestration Engine is created, that is fully FMI 2.0 compliant. This task corresponds to the “simulation and virtual integration” branch of the V in Figure 2.

To test systematically if a sub-system fulfills all its functional requirements, a test automation tool is used to create all relevant scenarios for the Co-simulation. In the INTO-CPS project, the RT Tester tool suite (www.verified.de/products/rt-tester) is used to create a FMU from the requirements that are defined in SysML. This FMU generates signals for the Co-Simulation to create the relevant test cases and analyzes the results. This represents another important part of the tool-chain in MBSE, where a SysML tool is coupled to a Test-automation tool, which then is used by a simulation tool.

While so far everything is done on the virtual level, generation of actual code that is executed on a hardware controller is the next step in the development cycle. All the modeling tools allow code generation. The hardware running its code is then connected to the rest of the system that is still virtual in a Hardware-in-the-loop scenario, again using the Co-Simulation Orchestration Engine to exchange the simulation signals. This task relates to the “integration and testing” branch of the V in Figure 2.

From the previous description, it is implicitly clear that many artifacts are created along the development cycle. These artifacts range from requirements to interface definitions, models, results, test-cases, code and more. All of these artifacts might exist in multiple versions and are created in different tools (themselves probably evolving in multiple versions) by multiple people. To master this complexity, and to answer questions such as “which requirements are related to this particular sub-model?”, or “which model versions were used to create this simulation result file?” traceability features are implemented in the INTO-CPS tool-chain. Here, traceability relies on the OSLC and Prov-N notations (www.w3.org/TR/prov-overview) for describing relations between artifacts.

One aspect that is of prime importance for the creation of such a tool-chain, and in particular for its openness, is the usage of standards. Here, the FMI standard is heavily used for simulation models and for exchange of signals between models. The OSLC and Prov-N standards are used to describe the traceability relations. The system architecture and specification is described by using the SysML standard.

By using open industrial standards as described, it is possible to combine various tools and process steps which are typical for the development of CPS. The shown steps allow or already use hooking on continuous integration mechanisms or coupling with companywide data-bases during the simulation and virtualization steps. This is essential to achieve on the one hand side correct results closing the development loops and on the other hand side for a consistent traceability. The details of such co-simulation and database coupling have been discussed already in literature [9].

5. CONCLUSION

The paper presented here shows how central and powerful graphs and Model-Based Systems Engineering (MBSE) are in the development process of Cyber Physical Systems (CPS). Complexity and short time frames for system development require traceability and verification of requirements and functional design, especially in dynamic domains such as the automotive or aerospace industry. MBSE can help to reduce costs and time of developments by providing consistent specifications. This allows for frontloading of subsequent development. It cuts down error rates and diminishes unnecessary interactions, such as setting up simulation with adapted boundary conditions due to different stages of specification. MSBE requires collaboration across different tools and methodologies, which can be reached by relying on standards. In this paper, this is demonstrated with the INTO-CPS project.

Therefore, the evaluation of MBSE approaches for the individual development process can be valuable. However, introducing MBSE is not just a technical issue, it also requires changes in processes and habits of the engineers.

6. ACKNOWLEDGEMENTS

The INTO-CPS project is funded by the Horizon2020 programme under Grant Agreement Number 644047.

7. REFERENCES

[1] ISO/IEC/IEEE 15288:2015; Systems and software engineering – System lifecycle processes (2015)

[2] J. Holt, S. Perry; SysML for Systems Engineering: A Model-Based Approach, Computing and Networks (2014)

[3] B. P. Douglass: Agile Systems Engineering, Elsevier / Morgan Kaufmann (2016)

[4] S. Friedenthal, A. Moore, R. Steiner; A practical guide to SysML, Morgan Kaufmann (2014)

[5] IBM Rational Harmony Deskbook Release 4.1 (2014)

[6] T. Weilkiens, SYSMOD – The Systems Modeling Toolbox – Pragmatic MBSE with SysML (2016)

[7] T. Weilkiens, J. Lamm, S. Roth, M. Walker; Model-based system architecture, Wiley (2015)

[8] P.G. Larsen et al, Integrated tool chain for model-based design of Cyber-Physical Systems: The INTO-CPS project, 2nd International Workshop on Modelling, Analysis, and Control of Complex CPS (CPS Data) (2016)

[9] U. Wurstbauer, M. Herrnberger, A. Raufeisen, V. Fäßler; Efficient Development of Complex Systems Using a Unified Modular Approach, Deutscher Luft- und Raumfahrtkongress (2015)